**Practical File**

**Object Oriented Programming**

**Using C++ (ITC 251)**

**B.E.(IT) 2nd Semester**

****

**Submitted To: Submitted By:**

Ms Ritika Pragya Sharma

UE228079

IT Section – 2

**DEPARTMENT OF INFORMATION TECHNOLOGY**

*University Institute of Engineering and Technology,*

*Panjab University,Chandigarh*

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.No** | **Program** | **Date** | **Remark** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1.Write a program in C++ to Print Hello World.**

**CODE:**

#include <iostream>

using namespace std;

int main()

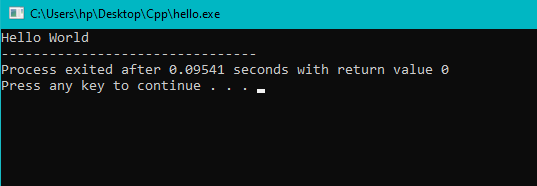
{

cout<<"Hello World";

return 0;

}

**Result:**



**2.Write a program in C++ to find whether number is even or odd**

**CODE**: -

#include <iostream>

using namespace std;

int main()

{

int n;

cout<<"TO CHECK WHETHER NUMBER IS EVEN OR ODD\n";

cout<<"Enter number n=";

cin>>n;

if( n%2 == 0)

cout<<"EVEN NUMBER";

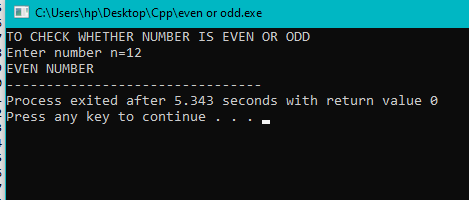
else

cout<<"ODD NUMBER";

return 0;

}

**OUTPUT:**

****

**3**.**Write a program in C++ to print sum of 2 numbers**.

**CODE: -**

#include <iostream>

using namespace std;

int main()

{

int a,b;

cout<<"SUM OF TWO NUMBERS";

cout<<"\nEnter the first number";

cin>>a;

cout<<"\nEnter the second number";

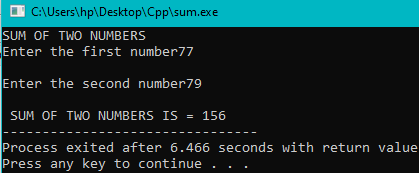
cin>>b;

cout<<"\n SUM OF TWO NUMBERS IS = "<<a+b;

return 0;

}

**OUTPUT:**



**4.** **Write a program to find factorial of a number**.

**Code:** -

#include <iostream>

using namespace std;

int main()

{

int x;

int fact =1;

cout<<"Factorial of a number \n";

cout<<"Enter the number \n";

cin>>x;

for(int i =x;i>0;i--)

{

fact =fact\*i;

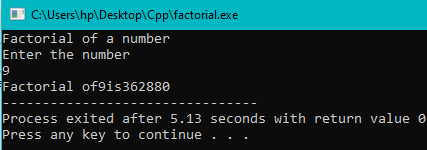
}

cout<<"Factorial of"<<x<<"is"<<fact;

return 0;

}

**OUTPUT:**



**5. Write a program to Print Fibonacci series.**

**Code: -**

#include <iostream>

using namespace std;

int main()

{

int n;

cout<<"FIBONACII SERIES \n";

cout<<"Enter the number n =";

cin>>n;

int a = 0 , b=1;

int sum = 1;

cout<<"\n"<<a;

cout<<"\n"<<b;

for( n;n>2;n--)

{

sum = a+b;

a=b;

b=sum;

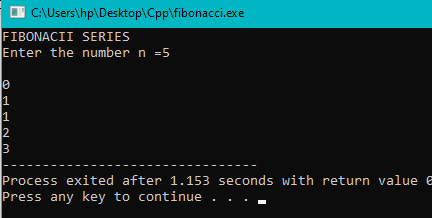
cout<<"\n"<<b;

}

return 0;

}

**OUTPUT:**



**6. Write a program to find greatest of 3 numbers.**

**CODE:**

#include <iostream>

using namespace std;

int main()

{

int a,b,c;

cout<<"Enter any three numbers";

cout<<"\nFirst number = ";

cin>>a;

cout<<"\nSecond number = ";

cin>>b;

cout<<"\nThird number = ";

cin>>c;

if(a>b && a>c)

{

cout<<"\n"<<a <<" is greatest";}

if(b>a && b>c)

{

cout<<"\n"<<b<<" is greatest";}

if(c>a && c>b)

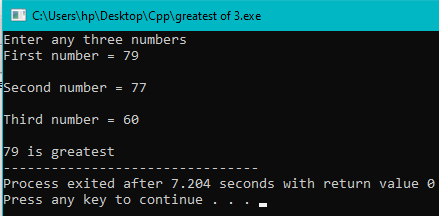
{

cout<<"\n"<<c<<" is greatest";}

return 0;

}

**OUTPUT:**

****

**7. Write a program to swap two variables without using third variable.**

**CODE:**

#include <iostream>

using namespace std;

int main()

{

int a,b;

cout<<"SWAPPING OF NUMBERS USING THIRD VARIABLE\n";

cout<<"Enter the first number ";

cin>>a;

cout<<"\n Enter the second number ";

cin>>b;

b=a+b;

a=b-a;

b=b-a;

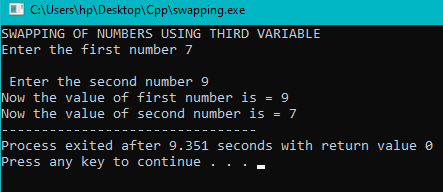
cout<<"Now the value of first number is = "<<a;

cout<<"\nNow the value of second number is = "<<b;

return 0;

}

**OUTPUT:**



**8. Write a program to check whether number is armstong or not.**

**CODE:**

#include <iostream>

using namespace std;

int main()

{

int x;

cout<<"ARMSTONG NUMBER \n";

cout<<"Enter the number = ";

cin>>x;

int s=0;

int m=x;

while(x>0)

{

int y = x%10;

s=s+(y\*y\*y);

x=x/10;

}

if(s==m)

{

cout<<m<<" IS A ARMSTONG NUMBER ";

}

else

{

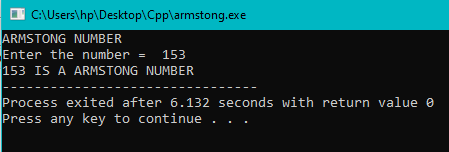
cout<<m<<"IS Not A ARMSTONG NUMBER";

}

return 0;

}

**OUTPUT:**

****

**9. Write a program in C++ to concatenate two strings.**

**CODE:**

#include <iostream>

#include <string>

using namespace std;

int main()

{

string Firstname = "Pragya";

string Lastname = "Sharma";

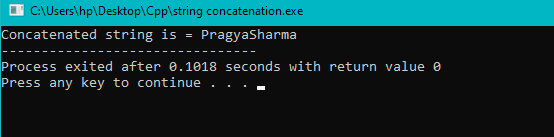
string Fullname = Firstname.append( Lastname);

cout<<"Concatenated string is = "<<Fullname;

return 0;

}

**OUTPUT:**

****

**10.Write a program to use public access specifier.**

**CODE:**

#include <iostream>

using namespace std;

class Circle

{

public:

double radius;

double area ()

{

if (radius >= 0)

{

return 3.14 \* radius \* radius;

}

else

{

cout<< "Area is not possible";

return 0;

}

}

};

int main ()

{

double r;

cout<<"AREA OF THE CIRCLE";

cout<<"\n Enter the radius = ";

cin>>r;

Circle obj;

obj.radius = r;

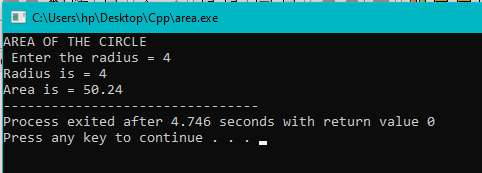
cout<< "Radius is = " <<obj.radius<< "\n";

cout<< "Area is = " <<obj.area ();

return 0;

}

**OUTPUT:**

****

**11.Write a program to use protected access specifier.**

**CODE:**

#include <iostream >

using namespace std;

class Parent{

protected:

int id\_protected;

};

class Child : public Parent{

public:

void setId(int id){

id\_protected = id;

}

void displayId() {

cout << "id\_protected is: " << id\_protected << endl;

} };

int main() {

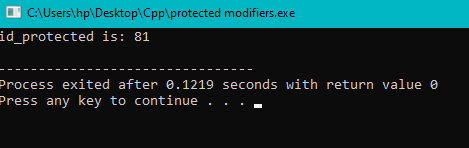
Child obj1;

obj1.setId(81);

obj1.displayId();

return 0;}

**OUTPUT:**

****

**12.Write a program using private access speifier.**

**CODE:**

#include <iostream>

using namespace std;

class circle

{

private:

double radius;

public:

void area(double r)

{

radius =r;

double area =3.14\*radius\*radius;

cout<<"Radius is ="<<radius;

cout<<"\n Area is ="<<area;

}

};

int main()

{

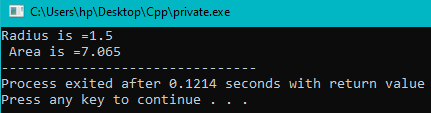
circle obj;

obj.area(1.5);

return 0;

}

**OUTPUT:**



**13.Write a program to show working of reference.**

**CODE:**

#include <iostream>

using namespace std;

int main()

{

int c =10;

int &r =c; //using reference variable

r =20;

cout<<"c = "<<r<<"\n";

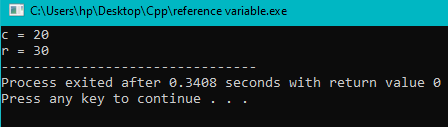
c=30;

cout<<"r = "<<r;

return 0;

}

**OUTPUT:**

****

**14.Write a program in C++to implement inline and static functions.**

**CODE:**

#include <iostream>

using namespace std;

inline int cube(int s){ return s\*s\*s;}

static int main()

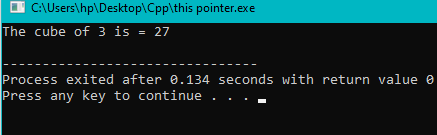
{

cout<<"The cube of 3 is = "<<cube(3)<<"\n";

return 0;

}

**OUTPUT:**

****

**15.Write a program in C++ to implement friend functions.**

**CODE:**

#include <iostream>

using namespace std;

class base {

private:

int private\_variable;

protected:

int protected\_variable;

public:

base()

{

private\_variable=19;

protected\_variable=98;

}

//Friend function definition

friend void friendfunction(base&Obj)

{

cout<<"Private variable : "<<Obj.private\_variable;

cout<<"\n Protected variable : "<<Obj.protected\_variable;

}

};

int main()

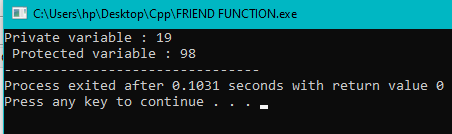
{

base object1;

friendfunction(object1);

return 0; }

**OUTPUT:**

****

**16.Write a program in C++ to implement array of objects.**

**CODE:**

#include <iostream>

using namespace std;

class Employee{

int id;

char name[30];

public:

void getdata();

void putdata();

};

void Employee::getdata(){

cout<<"Enter ID :";

cin>>id;

cout<<" Enter Name :";

cin>>name;

}

void Employee::putdata(){

cout<<id<<" ";

cout<<name<<"\n ";

cout<<endl;

}

int main()

{

Employee emp;

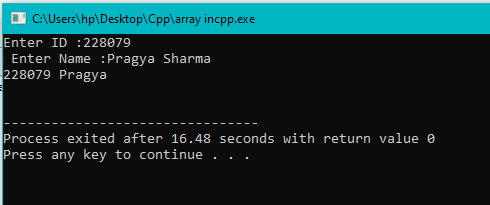
emp.getdata();

emp.putdata();

return 0;

}

**OUTPUT**

****

**17.Write a program in C++ to implement pointers to member functions.**

**CODE:**

#include <iostream>

using namespace std;

class AB{

public:

int sub(int a, int b){

return a-b;

}

int div(int a, int b){

return a/b;

}

};

//using function pointer

int res1(int m, int n, AB\* obj, int(AB::\*fp)(int,int)){

return(obj->\*fp)(m,n);

}

int res2(int m, int n, AB\* obj, int(AB::\*fp2)(int,int)){

return(obj->\*fp2)(m,n);

}

int main()

{

AB ob;

cout<<"Substraction is = "<<res1(8,5, &ob, &AB::sub)<<endl;

cout<<"Division is = "<<res2(4,2, &ob, &AB::div)<<endl;

return 0;

}

**OUTPUT:**

Text

Description automatically generated

**18.Write a program to implement types of constructors.**

**CODE:**

#include <iostream>

using namespace std;

class construct //example of constructor overloading

{

public:

float area;

construct() //default constructor

{

area = 0;

}

construct(int a, int b) //parametrised constructor

{

area = a\*b;

}

void display()

{

cout<<"Area is = " <<area <<endl;

}

};

int main()

{

construct o1;

construct o2(10 , 20);

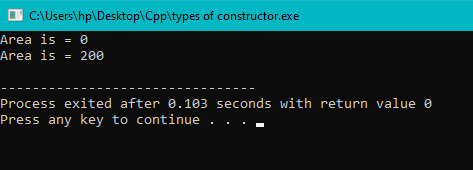
o1.display();

o2.display();

return 0;

}

**OUTPUT:**

****

**19.Write a program to implement the order of invocation of constructors and destructors.**

**CODE:**

#include <iostream>

using namespace std;

class Test{

public:

Test() { cout<<"CONSTRUTOR EXECUTED";}

~Test(){cout<<"\n DESTRUCTOR EXECUTED";}

};

int main()

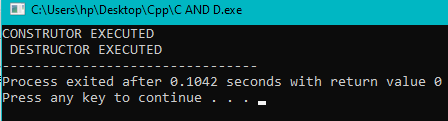
{

Test t;

return 0;

}

**OUTPUT:**

****

**20.Write a program in C++ to implement use of dynamic constructors.**

**CODE:**

#include <iostream>

using namespace std;

class geeks{

const char\* p;

public:

geeks()

{

//allocating memory at run time.

p = new char[6];

p="geeks";

}

void display()

{

cout<<p<<endl;

}

};

int main()

{

geeks obj1,obj2;

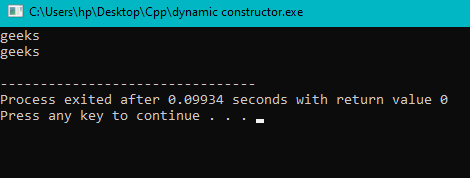
obj1.display();

obj2.display();

return 0;

}

**OUTPUT:**

****

**21.Write a program in C++ to implement function and operator overloading.**

**CODE:**

#include <iostream>

using namespace std;

class Box {

public:

double length;

double breadth;

double height;

Box(double l=0, double b=0, double h=0) {

cout<< "Box constructor called." <<endl;

length = l;

breadth = b;

height = h; }

double getVolume() {

return length \* breadth \* height; }

double getVolume(int factor) {

return length \* breadth \* height \* factor; }

Box operator+(const Box& b) {

Box box;

box.length = this->length + b.length;

box.breadth = this->breadth + b.breadth;

box.height = this->height + b.height;

return box; }};

int main() {

Box box1(2.0, 3.0, 4.0);

Box box2(5.0, 6.0, 7.0);

Box box3;

int factor;

cout<<"Enter the factor you want the volume to be multiplied : ";

cin>>factor;

cout<< "Volume of box1: " << box1.getVolume() <<endl;

cout<< "Volume of box1 with factor " << factor << ": " << box1.getVolume(factor) <<endl;

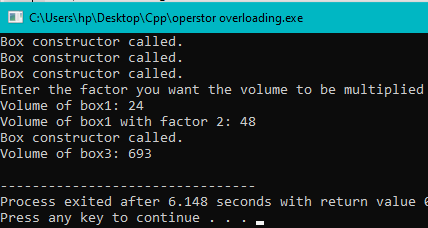
box3 = box1 + box2;

cout<< "Volume of box3: " << box3.getVolume() <<endl;

return 0;

}

**OUTPUT:**

****

**22.Write a program in C++ to implement overloading using friend function.**

**CODE:**

#include <iostream>

usingnamespacestd;

classDistance {

public:

    int feet, inch;

    Distance()

    {

        this->feet = 0;

        this->inch = 0;

    }

    Distance(int f, int i)

    {

        this->feet = f;

        this->inch = i;

    }

    friend Distance operator + (Distance&,

                                Distance&);

};

Distance operator+(Distance& d1,Distance& d2)

{  Distance d3;

    d3.feet = d1.feet + d2.feet;

    d3.inch = d1.inch + d2.inch;

  return d3;

}

  int main()

{

    Distance d1(8, 9);

    Distance d2(10, 2);

    Distance d3;

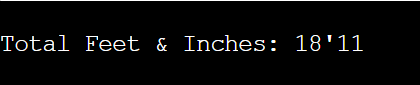
    d3 = d1 + d2;

  cout<<"\nTotal Feet & Inches: "<<d3.feet<<"'"<< d3.inch;

    return0;

}

**OUTPUT:**



**23. Write a program in C++ showing function overriding demonstrating run time polymorphism.**

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class animal //base class declaration

{

public:

string color = "BLACK";

};

class dog: public animal //inheriting animal class

{

public:

string color = "GREY";

};

int main()

{

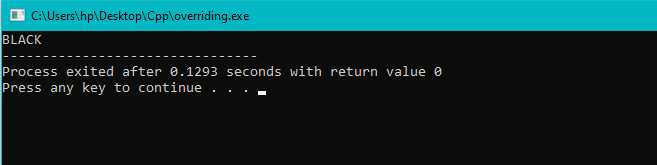
animal d = dog(); //accessing the field by reference variable

cout<<d.color;

return 0;

}

**OUTPUT:**

****

**24.Write a program to implement virtual functions.**

**CODE:**

#include <iostream>

using namespace std;

class base {

public:

virtual void print() {

cout << "print base class\n";

}

void show(){

cout << "show base class\n";

}

};

class derived : public base {

public:

void print(){

cout << "print derived class\n";

}

void show() {

cout << "show derived class\n";

}

};

int main(){

base \*bptr;

derived d;

bptr = &d;

// Virtual function, binded at runtime

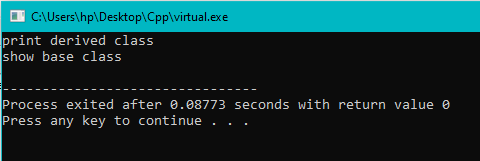
bptr->print();

// Non-virtual function, binded at compile time

bptr->show();

return 0;}

**OUTPUT:**

****

**25. Write a program in C++ to implement data conversions.**

**CODE:**

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

int main() {

string str = "123";

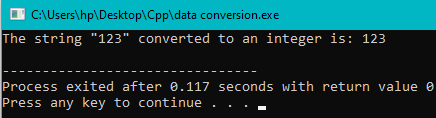
int num = atoi(str.c\_str());

cout<< "The string " << str << " converted to an integer is: " <<num<<endl;

return 0;

}

**OUTPUT:**

****

**26.Write a program showing various types of inheritance (Single, Multilevel inheritance, Hybrid inheritance)**

**CODE:**

#include <iostream>

using namespace std;

// base class

class Vehicle {

public:

Vehicle() { cout<< "This is a Vehicle\n"; }

};

// base class

class Fare {

public:

Fare() { cout<< "Fare of Vehicle\n"; }

};

// first sub class

class Car : public Vehicle {

};

// second sub class

class Bus : public Vehicle, public Fare {

};

// main function

int main()

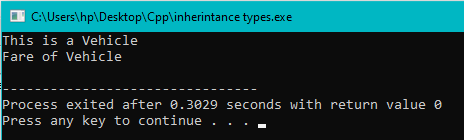
{

// Creating object of sub class will invoke the constructor of base class.

Bus obj2;

return 0;}

**OUTPUT:**

****

**27. Write a program in C++ to implement abstract classes.**

**CODE:**

#include <iostream>

using namespace std;

class base {

public:

virtual void print()

{

cout << "print base class\n";

}

void show()

{

cout << "show base class\n";

}

};

class derived : public base {

public:

void print()

{

cout << "print derived class\n";

}

void show()

{

cout << "show derived class\n";

}

};

int main()

{

base \*bptr;

derived d;

bptr = &d;

// Virtual function, binded at runtime

bptr->print();

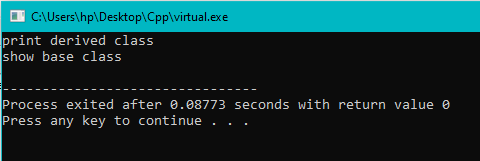
// Non-virtual function, binded at compile time

bptr->show();

return 0;

}

**OUTPUT:**

****

**28. Write a program in C++ to implement constant pointers.**

**CODE:**

#include<iostream>

using namespace std;

int main(){

int a{90};

int b{50};

int \* constptr(&a);

cout<<\*ptr<<"\n";

cout<<ptr<<"\n";

\*ptr=56;

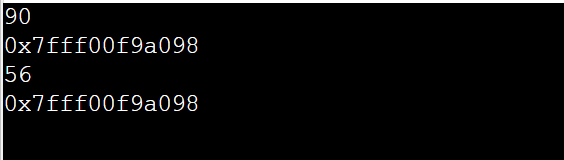
cout<<\*ptr<<"\n";

cout<<ptr<<"\n";

return 0;

}

**OUTPUT:**

****

**29. Write a program in C++ to implement pointers to derived and base classes.**

**CODE:**

#include<iostream>

using namespace std;

class Base

{

public:

virtual void show() { cout<<" In Base \n"; }

};

class Derived: public Base

{

public:

void show() { cout<<"In Derived \n"; }

};

int main(void)

{

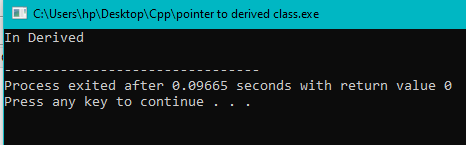
Base \*bp = new Derived;

bp->show(); // RUN-TIME POLYMORPHISM

return 0;

}

**OUTPUT:**

****

**30. Write a program to demonstrate the working of Abstraction.**

**CODE:**

#include<iostream>

using namespace std;

class implementAbstraction{

private:

int a,b;

public:

void set(int x,int y){

a=x;

b=y;

}

void display(){

cout<<"a= "<<a<<endl;

cout<<"b= "<<b<<endl;

}

};

int main(){

implementAbstraction obj;

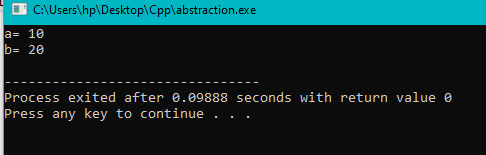
obj.set(10,20);

obj.display();

return 0;

}

**OUTPUT:**

****

**31.Write a program to implement encapsulation using classes and access modifiers.**

**CODE:**

#include<iostream>

using namespace std;

class encapsulation{

private:

int x;

public:

void set(int a){

x=a;

}

int get(){

return x;

}

};

int main(){

encapsulation obj;

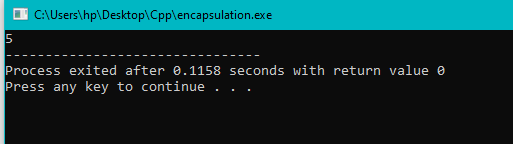
obj.set(5);

cout<<obj.get();

return 0;

}

**OUTPUT:**

****

**32. Write a program to read and write a File in C++(ifstream & ofstream).**

**CODE:**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

ofstream fout;

string line;

fout.open("sample.txt");

while(fout){

getline(cin,line);

if(line=="-1")

break;

fout<<line<<endl;

}

fout.close();

//creation of ifstream class object to read the file

ifstream fin;

fin.open("sample.txt");

while(getline(fin,line)){

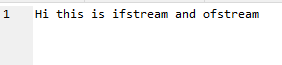
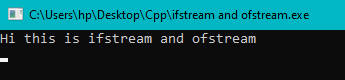
cout<<line<<endl;

}

fin.close();

}

**OUTPUT:**

****

**32>Write a program to read and write a File in C++(fstream).**

**Code:**

#include<iostream>

#include<fstream>

using namespace std;

int main(){

fstream fio;

string line;

fio.open("sample.txt",ios::trunc|ios::out|ios::in);

while(fio){

getline(cin,line);

if(line=="-1")

break;

fio<<line<<endl;

}

fio.seekg(0,ios::beg);

while(fio){

getline(fio,line);

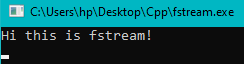
cout<<line<<endl;

}

fio.close();

}

**OUTPUT:**

** C:\Users\hp\Pictures\Screenshots\Screenshot (156).png**

**34.Write a program to represent Template working**

#include<iostream>

using namespace std;

template<class T>T add(T &a,T &b){

T result=a+b;

return result;

}

int main(){

int i=2;

int j=3;

float m=2.3;

float n=1.2;

cout<<"Addition of i and j is: "<<add(i,j);

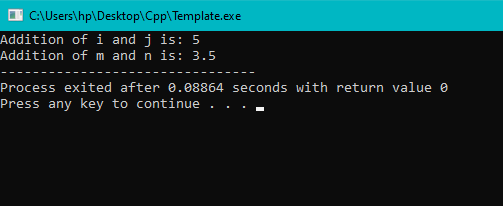
cout<<"\n";

cout<<"Addition of m and n is: "<<add(m,n);

return 0;

}

**OUTPUT:**

****

**35. Write a program to implement overloading a function template.**

**CODE:**

#include<iostream>

using namespace std;

template<class X>void fun(X a){

cout<<"value of a is: "<<a<<endl;

}

template<class X,class Y>void fun(X b,Y c){

cout<<"value of b is: "<<b<<endl;

cout<<"value of c is: "<<c<<endl;

}

int main(){

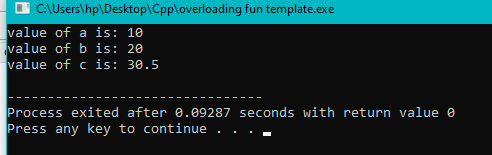
fun(10);

fun(20,30.5);

return 0;

}

**OUTPUT:**

****

**36. Write a program demonstrating exception handling using try, catch and throw.**

**CODE:**

#include <iostream>

using namespace std;

int main(){

int x = -1;

cout << "Before try \n";

try {

cout << "Inside try \n";

if (x < 0){

throw x;

cout << "After throw (Never executed) \n";

}

}

catch (int x ) {

cout << "Exception Caught \n";

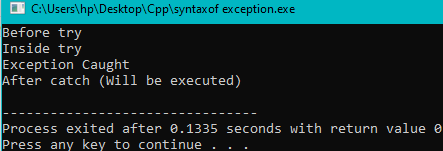
}

cout << "After catch (Will be executed) \n";

return 0;

}

**OUTPUT:**

**** **37. Write a program to implement exception handling.**

**CODE:**

#include<iostream>

using namespace std;

double division(int a,int b){

if(b==0){

throw "Division by 0 is not defined!!";

}

return (a/b);

}

int main(){

int x=50;

int y=0;

double z=0;

try{

z=division(x,y);

cout<<z<<endl;

}

catch(const char\* msg){

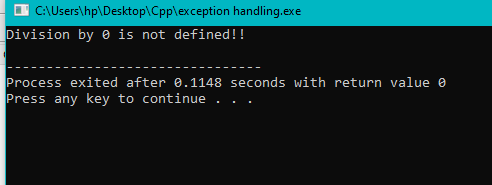
cerr<<msg<<endl;

}

return 0;

}

OUTPUT:



**38. Write a program in C++ to implement manipulators.**

**CODE:**

#include <cstdlib>

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

float number1;

number1 = 34.3358;

cout << number1 << endl;

cout << setprecision(2) << number1 << endl;

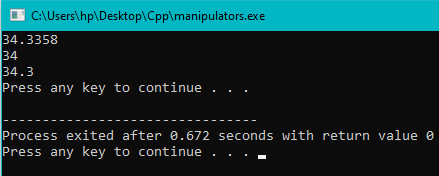
cout << setprecision(3) << number1 << endl;

system("PAUSE");

return EXIT\_SUCCESS;

}

**OUTPUT:**

****

**39. Write a program to implement user-defined manipulators.**

**CODE:**

#include <iostream>

#include <iomanip>

using namespace std;

ostream&curr(ostream&ostrObj)

{

cout << fixed << setprecision(2);

cout << "Rs.";

return ostrObj;

}

int main()

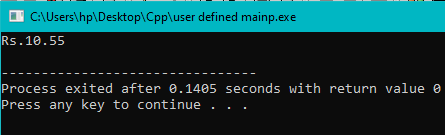
{

float amt = 10.5478;

cout << curr << amt <<endl;

}

**OUTPUT:**

****

**40. Write a program in C++ to implement Vector.**

**CODE:**

#include<iostream>

#include<vector>

using namespace std;

void display(vector<int> &v){

int i;

for(i=0;i<v.size();i++){

cout<<v[i]<<endl;

}

}

int main(){

vector <int> vec1;

int element,size;

cout<<"Enter size of vector: "<<endl;

cin>>size;

int i;

for(i=0;i<size;i++){

cout<<"Enter any element to add in this vector: "<<endl;

cin>>element;

//vector member function

vec1.push\_back(element);

}

vec1.pop\_back();

display(vec1);

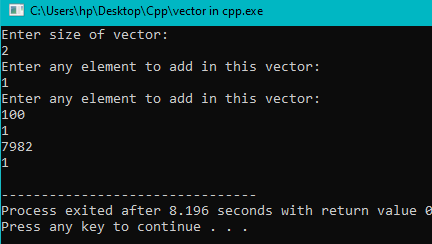
vector<int>::iterator it=vec1.begin();

vec1.insert(it,7982);

display(vec1);

}

**OUTPUT:**

****